Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1.1

з дисципліни
«Інтелектуальні вбудовані системи»

на тему
«ДОСЛІДЖЕННЯ І РОЗРОБКА МОДЕЛЕЙ ВИПАДКОВИХ СИГНАЛІВ. АНАЛІЗ ЇХ ХАРАКТЕРИСТИК»

Виконав:

студент групи ІП-84
Ковалишин Олег Юрійович
номер залікової книжки: 8410

Перевірив:

ас. кафедри ОТ
ас. Регіда П. Г.

Київ 2020

2. Теоретичні відомості

Випадковий сигнал або процес завжди представляється деякою функцією часу x(t) , значення якої не можна передбачити з точністю засобів вимірювання або обчислень, які б кошти моделі ми не використовували.

Для випадкового процесу його значення можна передбачити лише основні його характеристики: математичне сподівання Mx(t) , дисперсію Dx(t).

Ці характеристики для випадкового нестаціонарного процесу теж є функціями часу, але вони детерміновані. Для оцінки цих характеристик використовуються СРВ, які повинні обробити значну кількість інформації; для отримання їх при нестаціонарному процесі необхідно мати безліч реалізацій цього процесу.

Маточікування обчислюється за формулою

$$M_x = \lim_{N\to\infty} \frac{1}{N} \cdot \sum_{i=1}^{N} x_i(t_k) = \lim_{n\to\infty} \frac{1}{N} \sum_{k=0}^{n} x_i(t_k)$$

Дисперсія обчислюється за формулою:

$$D_x = \lim_{N\to 0} \cdot \frac{1}{N-1} \sum_{i=1}^{N} x_i(t_k) - M_x)^2 = \lim_{n\to\infty} \frac{1}{n-1} \cdot \sum_{k=0}^{n} (x_i(t_k) - M_x)^2 \geq 0$$

Генератор стаціонарного випадкового сигналу:

$$\overset{0}{x}(t) = \sum_{p=0}^{m} A_p \cdot \sin(w_p \cdot t + \varphi_p)$$

3. Умови завдання

Варіант 10:

n = 14, ω гр = 1700, N = 64

4. Вихідний код

```
import java.util.*
import kotlin.math.sin

data class Signal(val n: Int, val wMax: Int, val num: Int)

fun Signal.generate(): Array<Float> {
    val random = Random()
    val signals = Array(num) { 0f }
    for (i in 1..n) {
        val a = random.nextFloat()
        val fi = random.nextFloat()
        val w = wMax.toFloat() * i / n
        for (t in 0 until num) {
            val s = a * sin(w * t + fi)
            signals[t] += s
        }
    }

    return signals
}

import kotlinx.coroutines.*
import kscience.plotly.*
import kscience.plotly.models.XAnchor
import kscience.plotly.models.YAnchor
import kscience.plotly.palettes.Xkcd
import java.io.File
import kotlin.math.pow
import kotlin.system.measureTimeMillis

fun plotSignal(n: Int, wMax: Int, num: Int) {
    val signal = Signal(n, wMax, num)
```

```
    val records: Array<Float>
    val time = measureTimeMillis { records = signal.generate() }
    val m = records.average()
    val d = records.map { (it - m).pow(2) }.sum() / num
    println("Signal generated in $time msec, m = $m, D = $d")
    plot(
        records.indices, records.toList(),
        xAxis = "t",
        yAxis = "x(t)",
        clr = Xkcd.GREEN,
    )
}

fun benchmark(
    nMin: Int,
    nMax: Int,
    nStep: Int,
    wMax: Int,
    numMin: Int,
    numMax: Int,
    out: String? = null
) {
    val nTotal = (nMax - nMin) / nStep
    val numStep = (numMax - numMin) / nTotal
    val list = List(nTotal) { i ->
        GlobalScope.async {
            val n = nMin + i * nStep
            val num = numMin + i * numStep
            val time = measureTimeMillis { Signal(n, wMax, num).generate() }
            println("$i. Generated for n = $n N = $num, $time msec")
            time
        }
```

```
    }

    runBlocking {
        val time = list.awaitAll()
        plot(time.indices.mapIndexed { i, _ -> nMin + i * nStep }, time, "n", "msec")

        if (out != null) {
            val file = File(out)
            val d = ";"
            val csv = buildString {
                append("n" + d + "t")
                time.forEachIndexed { i, time -> append("\n$i$d$time") }
            }
            file.writeText(csv)
        }
    }
}

private fun plot(
    xData: Iterable<Number>,
    yData: Iterable<Number>,
    xAxis: String? = null,
    yAxis: String? = null,
    clr: String = Xkcd.BRIGHT_BLUE,
) {
    Plotly.page(mathJaxHeader, cdnPlotlyHeader) {
        plot {
            scatter {
                x.set(xData)
                y.set(yData)
                line { color(clr) }
            }
```

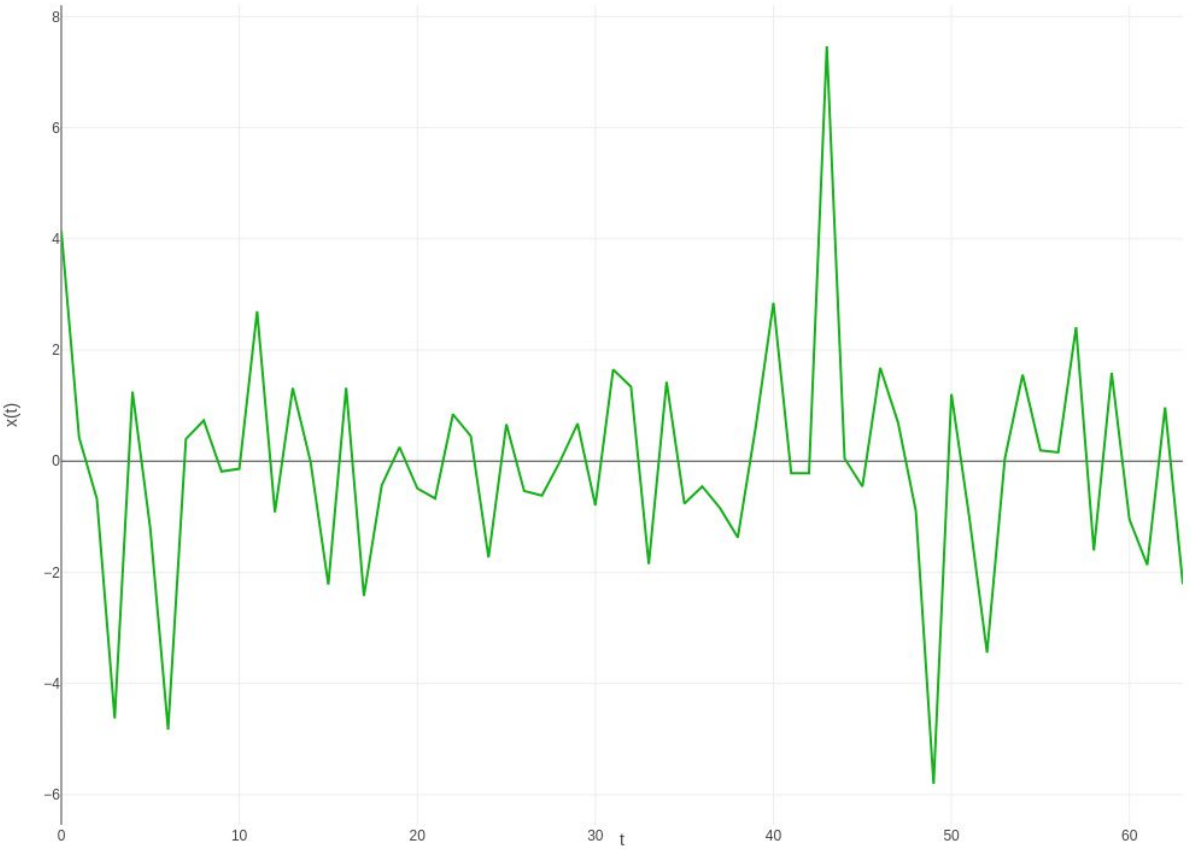
```
            layout {
                height = 750
                width = 1000
                margin { l = 50; r = 20; b = 20; t = 50 }
                xaxis.title = xAxis
                yaxis.title = yAxis
                legend {
                    x = 0.97
                    y = 1
                    borderwidth = 1
                    font { size = 32 }
                    xanchor = XAnchor.right
                    yanchor = YAnchor.top
                }
            }
        }
    }.makeFile()

}


fun main() {
    plotSignal(14, 1700, 256)
    benchmark(
        nMin = 1,
        nMax = 50_001,
        nStep = 5000,
        wMax = 1700,
        numMin = 1,
        numMax = 50_001,
        "res/out.csv"
    )
```

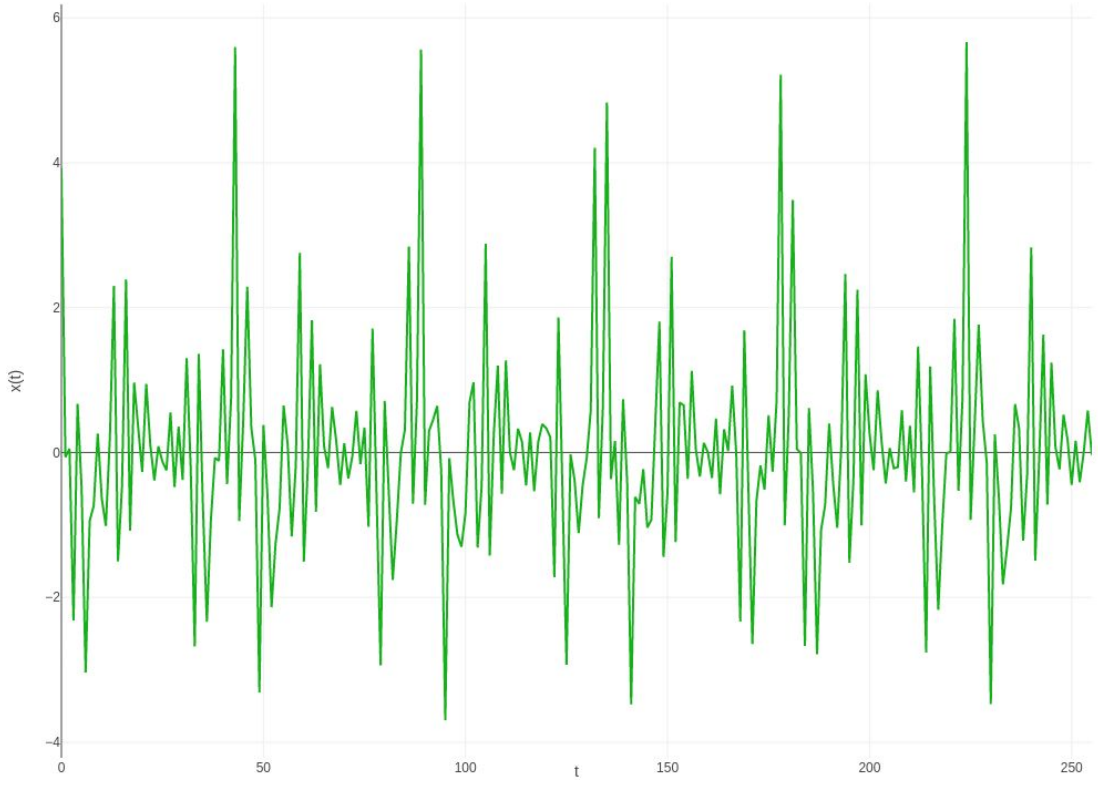
}

5. Результати виконання програми

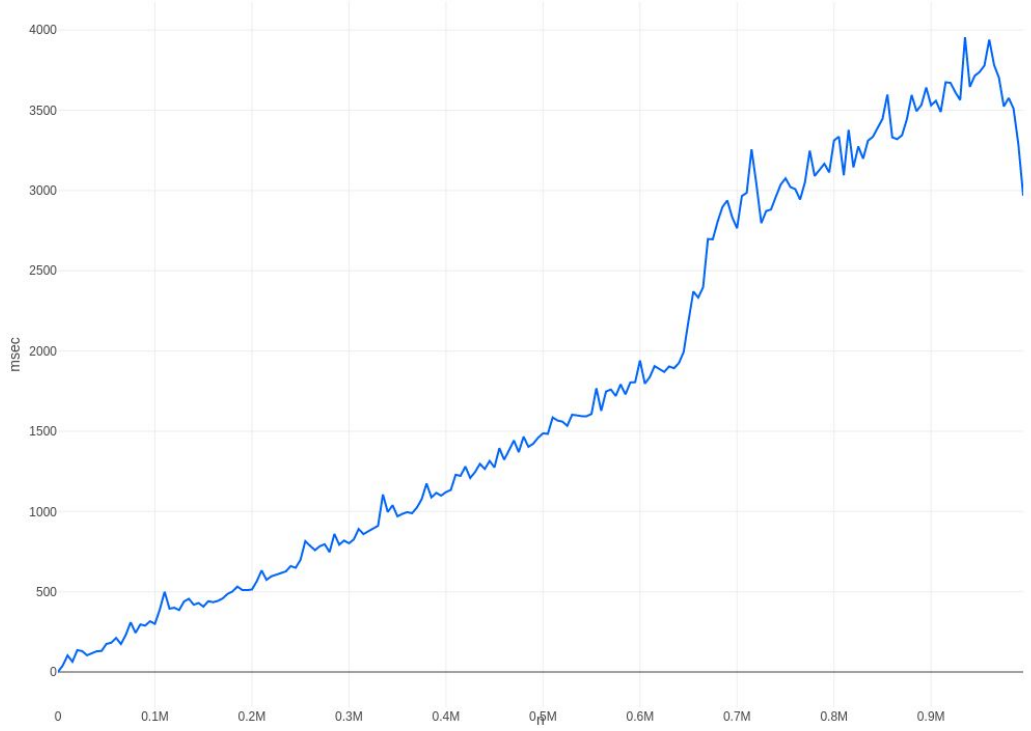


Сигнал, N = 64

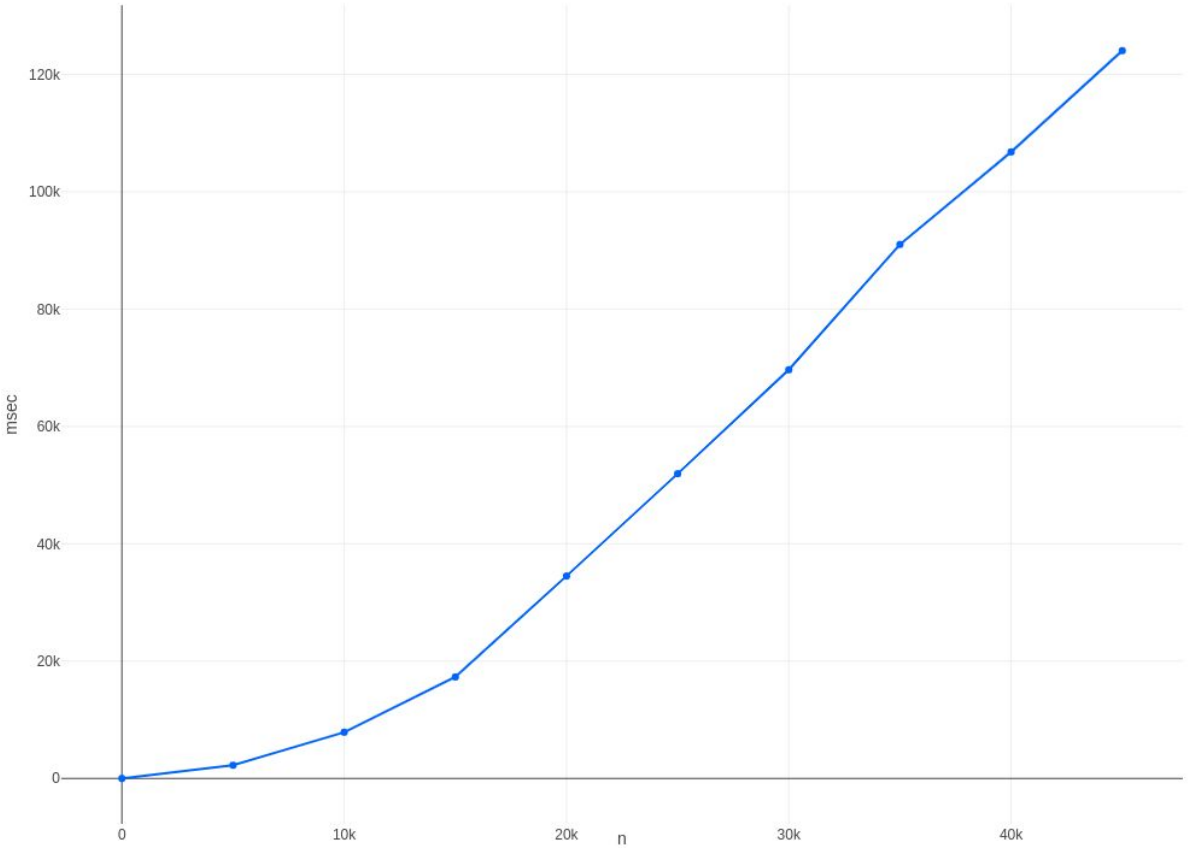
```
/home/alegator1209/.jdks/openjdk-15.0.2/bin/java ...
Signal generated in 2 msec, m = -0.009837845107540488, D = 2.5777796001562376
Відкривається в наявному сеансі веб-переглядача.
```



Сигнал, N = 256



Час обчислення сигналу, N стале, n змінюється.



Час обчислення сигналу, N і n змінюються, N = n

6. Висновки щодо виконання лабораторної роботи.

У ході виконання лабораторної роботи проведено ознайомлення з принципами генерації випадкових сигналів, вивчення та дослідження їх основних параметрів з використанням засобів моделювання і сучасних програмних оболонок.